Redirecionamentos em HTTP

Redirecionamento de URL, também conhecido como encaminhamento de URL, é uma técnica para atribuir mais de um endereço de URL a uma página, a um formulário, a um site inteiro ou a uma aplicação web.
O HTTP possui um tipo especial de resposta, chamado de **redirecionamento HTTP**, para essa operação.

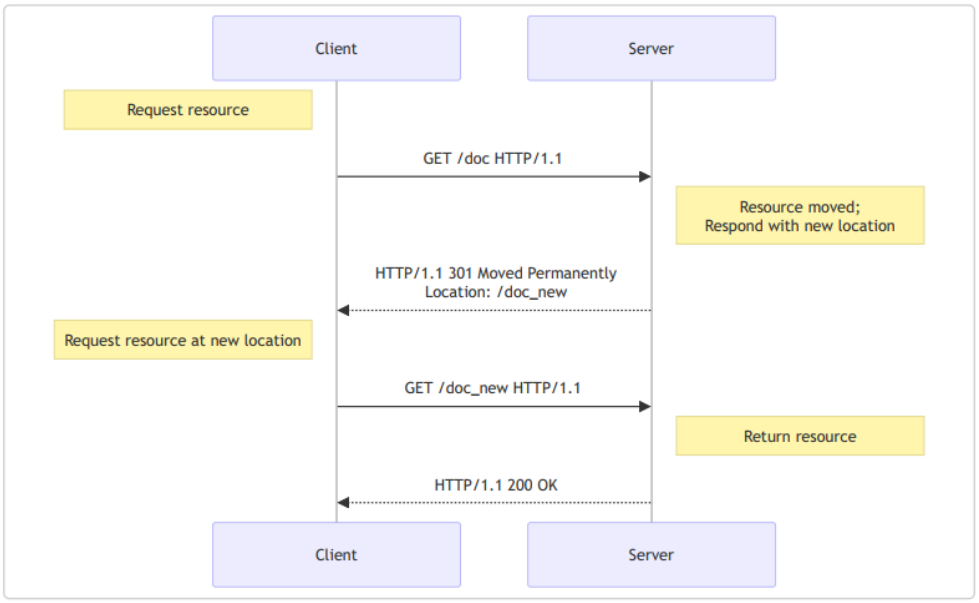
Redirecionamentos cumprem diversos objetivos:

- Redirecionamentos temporários durante manutenção ou inatividade do site
- Redirecionamentos permanentes para preservar links/favoritos existentes após mudar os URLs do site, páginas de progresso ao fazer upload de um arquivo, etc.

Princípio

No HTTP, o redirecionamento é acionado por um servidor que envia uma resposta especial de redirecionamento para uma requisição.
Respostas de redirecionamento possuem códigos de status que começam com **3**, e um cabeçalho **Location** contendo a URL para onde redirecionar.

Quando navegadores recebem um redirecionamento, eles carregam imediatamente a nova URL fornecida no cabeçalho Location.
Além de um pequeno impacto de desempenho devido à ida e volta adicional, os usuários raramente percebem o redirecionamento.



Tipos de redirecionamento

Existem vários tipos de redirecionamentos, classificados em três categorias:

1. Redirecionamentos permanentes
2. Redirecionamentos temporários
3. Redirecionamentos especiais

Redirecionamentos permanentes

Esses redirecionamentos são feitos para durar para sempre. Eles indicam que a URL original não deve mais ser usada e deve ser substituída pela nova.
Robôs de motores de busca, leitores de RSS e outros rastreadores atualizarão a URL original do recurso.

Código	Texto	Tratamento de método	Caso típico
301	Moved Permanently	Métodos GET permanecem inalterados. Outros podem ou não ser alterados para GET. [1]	Reorganização de um site.
308	Permanent Redirect	Método e corpo não são alterados.	Reorganização de um site com operações não-GET.

[1] A especificação não pretendia permitir a mudança de método, mas há agentes de usuário existentes que de fato mudam. O código **308** foi criado para remover a ambiguidade do comportamento ao usar métodos diferentes de GET.

Redirecionamentos temporários

Às vezes, o recurso requisitado não pode ser acessado de sua localização canônica, mas pode ser acessado de outro local. Nesse caso, um redirecionamento temporário pode ser usado.

Robôs de motores de busca e outros rastreadores **não memorizam** a nova URL temporária.
Redirecionamentos temporários também são usados ao criar, atualizar ou excluir recursos, para mostrar páginas de progresso temporárias.

Código	Texto	Tratamento de método	Caso típico
302	Found	Métodos GET permanecem inalterados. Outros podem ou não ser alterados para GET. [2]	A página da web está temporariamente indisponível por motivos imprevistos.
303	See Other	Métodos GET permanecem inalterados. Outros **são** alterados para GET (corpo é perdido).	Usado para redirecionar após um PUT ou POST, de forma que atualizar a página de resultado não reative a operação.
307	Temporary Redirect	Método e corpo não são alterados.	Página da web está temporariamente indisponível. Melhor que 302 quando há operações não-GET no site.

[2] A especificação não pretendia permitir a mudança de método, mas há agentes de usuário existentes que de fato mudam. O código **307** foi criado para remover a ambiguidade do comportamento com métodos não-GET.

Redirecionamentos especiais

- **304 (Not Modified)** redireciona uma página para a cópia em cache local (que estava obsoleta)
- **300 (Multiple Choices)** é um redirecionamento manual: o corpo, apresentado pelo navegador como uma página da web, lista as possíveis opções e o usuário clica em uma para selecionar.

Código	Texto	Caso típico
300	Multiple Choices	Poucos casos: as escolhas são listadas em uma página HTML no corpo. É recomendado que escolhas legíveis por máquina sejam enviadas como cabeçalhos `Link` com `rel=alternate`.
304	Not Modified	Enviado para requisições condicionais revalidadas. Indica que a resposta em cache ainda está atual.

Outra forma de especificar redirecionamentos

Redirecionamentos HTTP não são a única maneira de definir redirecionamentos. Existem outras duas:

1. Redirecionamentos HTML com o elemento <meta>

2. Redirecionamentos JavaScript via DOM

Redirecionamentos HTML

Redirecionamentos HTTP são a melhor maneira de criar redirecionamentos, mas às vezes você não tem controle sobre o servidor.
Nesse caso, tente usar um elemento <meta> com o atributo http-equiv definido como Refresh no <head> da página. Ao exibir a página, o navegador irá para a URL indicada.

O atributo content deve começar com um número indicando quantos segundos o navegador deve esperar antes de redirecionar para a URL fornecida.
Sempre defina como 0 para conformidade com acessibilidade.

Obviamente, esse método só funciona com HTML e não pode ser usado para imagens ou outros tipos de conteúdo.

```html
<head>
  <meta http-equiv="Refresh" content="0; URL=https://example.com/" />
</head>
```

Redirecionamentos JavaScript

Redirecionamentos em JavaScript são realizados atribuindo uma string de URL à propriedade window.location, carregando a nova página:

```javascript
window.location = "https://example.com/";
```

Assim como os redirecionamentos HTML, isso **não funciona com todos os recursos**, e obviamente **só funcionará em clientes que executam JavaScript**.
Por outro lado, existem mais possibilidades: por exemplo, você pode acionar o redirecionamento **somente se certas condições forem atendidas**.

Ordem de precedência

Com três formas de acionar redirecionamentos, vários métodos podem ser usados ao mesmo tempo. Mas qual é aplicado primeiro?

1. Redirecionamentos **HTTP** sempre são executados primeiro — eles existem antes mesmo de haver uma página transmitida.
2. Surpreendentemente, redirecionamentos **JavaScript** são executados em seguida, antes dos redirecionamentos HTML.
   Isso ocorre porque o redirecionamento <meta> acontece **depois que a página está completamente carregada**, o que ocorre **após todos os scripts terem sido executados**.
3. Redirecionamentos **HTML (<meta>)** são executados **somente se** não houver redirecionamentos HTTP ou JavaScript executados antes do carregamento da página.
4. Se houver qualquer redirecionamento JavaScript **após** o carregamento da página (por exemplo, ao clicar em um botão), ele será executado **por último**, desde que a página não tenha sido redirecionada anteriormente por outros métodos.

Quando possível, use redirecionamentos HTTP e **não adicione redirecionamentos via <meta>**.
Se alguém alterar os redirecionamentos HTTP mas esquecer de alterar os redirecionamentos HTML, eles deixarão de ser idênticos, o que pode causar **um loop infinito ou outros pesadelos**.

Casos de uso

Há inúmeros casos de uso para redirecionamentos, mas como o desempenho é impactado a cada redirecionamento, seu uso deve ser mantido **no mínimo necessário**.

Alias de domínio

Idealmente, existe uma única localização, e portanto uma única URL, para cada recurso. Mas há razões para nomes alternativos para um recurso:

Um caso comum é quando um site está em www.exemplo.com, mas acessá-lo via exemplo.com também deve funcionar.
Redirecionamentos de exemplo.com para www.exemplo.com são, portanto, configurados.
Você também pode redirecionar a partir de sinônimos comuns ou erros de digitação frequentes de seus domínios.

Por exemplo, sua empresa foi renomeada, mas você deseja que links ou favoritos existentes **ainda consigam encontrá-lo** com o novo nome.

Forçando HTTPS

Requisições para a versão http:// do seu site redirecionarão para a versão https:// do seu site.

Mantendo links ativos

Quando você reestrutura sites, URLs mudam.
Mesmo que você atualize os links do seu site para corresponder às novas URLs, você **não tem controle sobre os links externos**.

Você não quer quebrar esses links, pois eles trazem usuários valiosos e ajudam no SEO, então você configura redirecionamentos das URLs antigas para as novas.

Expandindo o alcance do seu site

Mudando para um novo domínio.

Nota: Essa técnica **funciona para links internos**, mas tente evitar redirecionamentos internos.
Um redirecionamento tem um custo de desempenho significativo (pois uma requisição HTTP extra ocorre).
Se você puder evitá-lo corrigindo os links internos, **corrija-os**.

Respostas temporárias para requisições inseguras

Requisições **inseguras** modificam o estado do servidor, e o usuário **não deveria reenviá-las** de forma não intencional.

Normalmente, você não deseja que seus usuários reenviem requisições **PUT**, **POST** ou **DELETE**.
Se você servir a resposta como resultado direto dessa requisição, um clique no botão de recarregar irá **reenviar a requisição** (possivelmente após uma mensagem de confirmação).

Nesse caso, o servidor pode retornar uma resposta **303 (See Other)** apontando para uma URL que conterá as informações corretas.
Se o botão de recarregar for pressionado, **apenas essa página é recarregada**, sem repetir a requisição insegura.

Respostas temporárias para requisições demoradas

Algumas requisições podem demandar mais tempo no servidor, como requisições **DELETE** que são agendadas para processamento posterior.
Nesse caso, a resposta é um redirecionamento **303 (See Other)** que aponta para uma página indicando que a ação foi agendada, e que eventualmente informa o progresso da operação ou permite cancelá-la.

Configurando redirecionamentos em servidores comuns

Apache

Redirecionamentos podem ser definidos tanto no **arquivo de configuração do servidor** quanto no arquivo **.htaccess** de cada diretório.

O módulo mod_alias possui as diretivas Redirect e RedirectMatch que, por padrão, configuram redirecionamentos **302**:

```apacheconf
<VirtualHost *:443>
  ServerName example.com
  Redirect / https://www.example.com
</VirtualHost>
```

A URL https://example.com/ será redirecionada para https://www.example.com/, assim como qualquer arquivo ou diretório abaixo dela
(ex: https://example.com/alguma-pagina será redirecionada para https://www.example.com/alguma-pagina)

RedirectMatch faz o mesmo, mas utiliza uma **expressão regular** para definir um conjunto de URLs afetadas:

```apacheconf
RedirectMatch ^/images/(.*)$ https://images.example.com/$1
```

Todos os documentos no diretório images/ serão redirecionados para um domínio diferente.

Se você **não quiser** um redirecionamento temporário, um parâmetro extra (o código de status HTTP ou a palavra-chave permanent) pode ser usado para configurar outro tipo de redirecionamento:

```apacheconf
Redirect permanent / https://www.example.com
# …atua da mesma forma que:
Redirect 301 / https://www.example.com
```

O módulo mod_rewrite também pode criar redirecionamentos. Ele é mais flexível, porém um pouco mais complexo.

Nginx

No **Nginx**, você cria um **bloco de servidor** específico para o conteúdo que deseja redirecionar:

```nginx
server {
  listen 80;
  server_name example.com;
  return 301 $scheme://www.example.com$request_uri;
}
```

Para aplicar um redirecionamento a um diretório ou apenas a certas páginas, use a diretiva rewrite:

```nginx
rewrite ^/images/(.*)$ https://images.example.com/$1 redirect;
rewrite ^/images/(.*)$ https://images.example.com/$1 permanent;
```

IIS

No **IIS**, você usa o elemento <httpRedirect> para configurar redirecionamentos.

Loops de redirecionamento

Loops de redirecionamento ocorrem quando redirecionamentos adicionais seguem um que já foi seguido.
Em outras palavras, há um **ciclo que nunca será concluído**, e nenhuma página será encontrada.

Na maioria das vezes, isso é um problema de configuração do servidor.
Se o servidor puder detectá-lo, ele retornará um erro **500 Internal Server Error**.
Se você encontrar esse erro logo após modificar uma configuração de servidor, provavelmente é um **loop de redirecionamento**.

Às vezes, o servidor **não detectará** o problema: um loop de redirecionamento pode se estender por **vários servidores**, e nenhum deles terá a imagem completa.
Nesse caso, os navegadores detectam o problema e exibem uma mensagem de erro.

O **Firefox** exibe:

O Firefox detectou que o servidor está redirecionando a solicitação para este endereço de forma que nunca será concluída.

O **Chrome** exibe:

Esta página da Web tem um loop de redirecionamento

Em ambos os casos, o usuário **não pode fazer muita coisa** (a menos que haja corrupção do lado dele, como conflito de cache ou cookies).

É importante evitar **loops de redirecionamento**, pois eles **quebram completamente a experiência do usuário**.